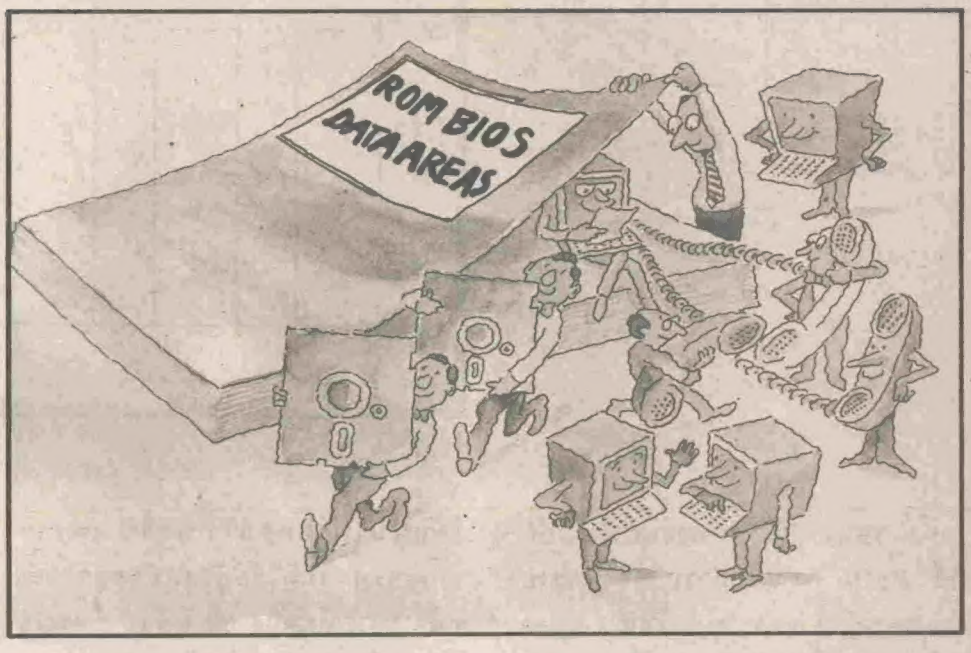
บุญแสง สุรวัฒนบุตร

การอ่าน ROM BIOS DATA AREAS

ตัวแปรแสดงสถานะของเครื่อง เพื่อนำไปประยุกต์โปรแกรม ควบคุมการใช้งาน ดูรายละเอียดของเครื่องได้



บางครั้งในการพัฒนาโปรแกรม เราไม่สามารถทำให้โปรแกรมทำงานบนเครื่องได้ทุกประเภท ดังนั้น การที่โปรแกรมสามารถอ่านสถานภาพของเครื่อง โดยที่ผู้ใช้ไม่จำเป็นต้องทำการป้อน จะทำให้การทำงานของโปรแกรมมีความสมบูรณ์ยิ่งขึ้น การอ่านค่าสถานภาพของเครื่องพีซี อย่างเช่น เป็นเครื่องประเภทใด เอกซ์ที หรือ เอที มีหน่วยความจำหลักของเครื่องเท่าใด จำนวนของดิสค์ไดรฟ์ที่มีอยู่ หรือการอ่านค่าของคีย์ต่าง ๆ เช่น NumLock ถูกเซตอยู่หรือไม่ ตลอดจนข้อมูลอื่น ๆ ของระบบที่ทำงานอยู่

บทความนี้จะครอบคลุมการแสดงข้อมูลในรอมไบออสดาต้าเอเรียส์ ที่น่าสนใจหลายตัวด้วยกัน เพื่อให้ผู้อ่านสามารถนำไปใช้งานได้ โดยเราจะเริ่มจากรูปที่ 1 ซึ่งแสดงรอมไบออสลิสติ้งจาก PC/AT Technical Reference ในรูปจะแสดงเฉพาะส่วนที่เป็นรอมไบออสดาต้าเอเรียส์ ซึ่งอยู่ระหว่างหน้า 5-29 ถึง 5-31 สิ่งแรกที่ต้องทำในการอ่านค่าคือ การสร้างแบบของข้อมูลให้มีลักษณะเหมือนข้อมูลที่จะอ่าน ดังนั้น แบบของข้อมูล BiosDataType ในบรรทัดที่ 17-48 ของยูนิตไบออสในรูปที่ 2 จึงถูกสร้างตามรอมไบออสลิสติ้ง สังเกตการตั้งชื่อฟิลด์ในเรคอร์ดของตัวแปรจะเลียนแบบจากรอมไบออสลิสติ้ง ตัวแปรที่จะอ่านเพื่อนำมาใช้เพื่ออ่านสถานภาพของเครื่องมีดังนี้

รูปที่ 1 รอมไบออสลิสติ้ง

```
;--------------------------------------------------------------------
; 0286 INTERRUPT LOCATIONS (RLAD):
;--------------------------------------------------------------------
0000                C ABSO SEGMENT AT 0
0000                C STC_LOCO                  LABEL BYTE
0008                C           ORG 2*4
0008                C V NMI_PTR                 LABEL WORD
0014                C           ORG 5*4
0014                C INT5_PTR                  LABEL WORD
0020                C           ORG 8*4
0020                C INT_ADDR                  LABEL WORD
0020                C INT_PIR                   LABEL DWORD
0040                C           ORG 10H*4
0040                C VIDEO_INT                 LABEL DWORD
0040                C           ORG 13H*4                   ; NEW FDISK
0040                C ORG_VECTOR                LABEL DWORD
```

```
0060                    C       ORG 18H*4
0060                    C BASIC_PTR             LABEL DWORD
0064                    C       ORG 19H*4
0064                    C BOOT_VEC              LABEL DWORD
0064                    C BOOT_VECTOR           LABEL DWORD
0074                    C       ORG 1DH*4
0074                    C PARM_PTR              LABEL DWORD ; POINTER TO VIDEO PARMS
0078                    C       ORG 1EH*4
0078                    C DISK_POINTER          LABEL DWORD
0070                    C       ORG 01FH*4
0070                    C EXT_PTR               LABEL DWORD
0100                    C       ORG 40H*4               ; DISKETTE POINTER
0100                    C DISK_VECTOR           LABEL DWORD
0104                    C       ORG 41H*4
0104                    C HF_TBL_VEC            LABEL DWORD
0118                    C       ORG 46H*4
0118                    C HF1_TBL_VEC           LABEL DWORD
0100                    C       ORG 70H*4
0100                    C SLAVE_INT_PTR         LABEL DWORD
0100                    C RTC_INT_VEC           LABEL DWORD ; REAL TIME CLOCK INT
0108                    C       ORG 76H*4               ; FIXED DISK INTERRUPT VECTOR
0108                    C HDISK_INT             LABEL DWORD
0400                    C       ORG     400H
0400                    C DATA_AREA             LABEL BYTE  ; ABSOLUTE LOCATION OF DATA SEGMENT
0400                    C DATA_WORD             LABEL WORD
0500                    C       ORG     0500H
0500                    C MFG_TEST_RTN          LABEL FAR
7000                    C       ORG     7000H
7000                    C BOOT_LOCN             LABEL FAR
7000                    C ABS0  ENDS
                        C PAGE
                        C ;--------------------------------------------
                        C ; STACK -- USED DURING INITIALIZATION ONLY:
                        C ;--------------------------------------------
0000                    C STACK SEGMENT AT 30H
0000 80 [               C       DW      128 DUP (?)
         ????           C
                      ] C
0100                    C TOS   LABEL   WORD
0100                    C STACK ENDS
                        C ;--------------------------------------------
                        C ;         ROM BIOS DATA AREAS:
                        C ;--------------------------------------------
0000                    C DATA  SEGMENT AT 40H
0000                    C DATA_BASE             LABEL BYTE
0000 04 [               C RS232_BASE            DW      4 DUP(?)    ; ADDRESSES OF RS232 ADAPTERS
         ????           C
                      ] C
                        C
0008 04 [               C PRINTER_BASE          DW      4 DUP(?)    ; ADDRESSES OF PRINTERS
         ????           C
                      ] C
                        C
0010 01 [               C EQUIP_FLAG            DW      1 DUP(?)    ; INSTALLED HARDWARE
         ????           C
                      ] C
0012 01 [               C MFG_TST               DB      1 DUP(?)    ; INITIALIZATION FLAG
         ??             C
                      ] C
0013 01 [               C MEMORY_SIZE           DW      1 DUP(?)    ; MEMORY SIZE IN K BYTES
         ????           C
                      ] C
0015 01 [               C MFG_ERR_FLAG          DB      1 DUP(?)    ; SCRATCHPAD FOR MANUFACTURING
         ??             C
                      ] C
0016 01 [               C                       DB      1 DUP(?)    ; ERROR CODES
         ??             C
                      ] C
                        C PAGE
                        C ;--------------------------------------------
                        C ; KEYBOARD DATA AREAS:
                        C ;--------------------------------------------
0017 01 [               C KB_FLAG               DB      1 DUP(?)
         ??             C
                      ] C
0018 01 [               C KB_FLAG               DB      1 DUP(?)    ; SECOND BYTE OF KEYBOARD STATUS
         ??             C
                      ] C
0019 01 [               C ALT_INPUT             DB      1 DUP(?)    ; STORAGE FOR ALTERNATE KEYPAD ENTRY
         ??             C
                      ] C
001A 01 [               C BUFFER_HEAD           DW      1 DUP(?)    ; POINTER TO HEAD OF KEYBOARD BUFFER
         ????           C
                      ] C
001C 01 [               C BUFFER_TAIL           DW      1 DUP(?)    ; POINTER TO TAIL OF KEYBOARD BUFFER
         ????           C
                      ] C
001E 10 [               C KB_BUFFER             DW      16 DUP(?)   ; ROOM FOR 15 ENTRIES
         ????           C
                      ] C
003E                    C KB_BUFFER_END         LABEL WORD
                        C ;--------HEAD=TAIL INDICATES THAT THE BUFFER IS EMPTY
                        C ;--------------------------------------------
                        C ; DISKETTE DATA AREAS:
                        C ;--------------------------------------------
003E 01 [               C SEEK_STATUS           DB      1 DUP(?)    ; DRIVE RECAL:BRATION STATUS
         ??             C
                      ] C
                        C                                           ; BIT 3-0 = DRIVE 3-0 NEEDS RECAL
                        C                                           ; BEFORE NEXT SEEK IF BIT IS = 0
003F 01 [               C MOTOR_STATUS          DB      1 DUP(?)    ; MOTOR STATUS
         ??             C
                      ] C
                        C                                           ; BIT 3-0 = DRIVE 3-0 IS CURRENTLY
                        C                                           ;   RUNNING
                        C                                           ; BIT / = CURRENT OPERATION IS A WRITE,
                        C                                           ;   REQUIRES DELAY
0040 01 [               C MOTOR_COUNT           DB      1 DUP(?)    ; TIME OUT COUNTER FOR DRIVE TURN OFF →
```

รูปที่ 1 (ต่อ)

```
        ??      C
          ]     C
0041 01 [       C DISKETTE_STATUS  DB    1 DUP(?)    ; RETURN CODE STATUS BYTE
        ??      C
          ]     C
0042            C CMD_BLOCK        LABEL BYTE
0042            C HD_ERROR         LABEL BYTE
0041 0/ [       C NEC_STATUS       DB    7 DUP(?)    ; STATUS BYTES FROM NEC
        ??      C
          ]     C
                C PAGE
                C ;-----------------------------
                C ; VIDEO DISPLAY DATA AREA:
                C ;-----------------------------
0049 01 [       C CRT_MODE         DB    1 DUP(?)    ; CURRENT CRT MODE
        ??      C
          ]     C
004A 01 [       C CRT_COLS         DW    1 DUP(?)    ; NUMBER OF COLUMNS ON SCREEN
        ????    C
          ]     C
004C 01 [       C CRT_LEN          DW    1 DUP(?)    ; LENGTH OF REGEN IN BYTES
        ????    C
          ]     C
004E 01 [       C CRT_START        DW    1 DUP(?)    ; STARTING ADDRESS IN REGEN BUFFER
        ????    C
          ]     C
0050 08 [       C CURSOR_POSN      DW    8 DUP(?)    ; CURSOR FOR EACH OF UP TO 8 PAGES
        ????    C
          ]     C
0060 01 [       C CURSOR_MODE      DW    1 DUP(?)    ; CURRENT CURSOR MODE SETTING
        ????    C
          ]     C
0062 01 [       C ACTIVE_PAGE      DB    1 DUP(?)    ; CURRENT PAGE BE NO DISPLAY
        ??      C
          ]     C
0063 01 [       C ADDR_6845        DB    1 DUP(?)    ; BASE ADDRESS FOR ACTIVE DISPLAY CARD
        ????    C
          ]     C
0065 01 [       C CRT_MODE_SET     DB    1 DUP(?)    ; CURRENT SETTING OF THE 3x8 REGISTER
        ??      C
          ]     C
0066 01 [       C CRT_PALLETTE     DB    1 DUP(?)    ; CURRENT PALLETTE SETTING COLOR CARD
        ??      C
          ]     C
                C PAGE
                C ;-----------------------
                C ; POST DATA AREA:
                C ;-----------------------
0067  01 [       C IO_ROM_INIT     DW    1 DUP(?)    ; PNIR TO  OPTIONAL  I/O ROM INIT ROUTINE
        ????    C
          ]     C
0069 01 [       C IO_ROM_SEG       DW    1 DUP(?)    ; POINTER TO 10 ROM.SEGMENT
        ????    C
          ]     C
006B 01 [       C INTR_FLAG        DB    1 DUP(?)    ; FLAG TO INDICATE AN INTERRUPT HAPPEND
        ??      C
          ]     C
                C ;-----------------------
                C ; TIMER DATA AREA:
                C ;-----------------------
006C 01 [       C TIMER_LOW        DW    1 DUP(?)    ; LOW WORD OF TIMER COUNT
        ????    C
          ]     C
006E 01 [       C TIMER_HIGH       DW    1 DUP(?)    ; HIGH WORD OF TIMER COUNT
        ????    C
          ]     C
0070 01 [       C TIMER_OFL        DB    1 DUP(?)    ; TIMER HAS ROLLED OVER SINCE LAST READ
        ??      C
          ]     C
                C ;-----------------------
                C ; SYSTEM DATA AREA:
                C ;-----------------------
0071 01 [       C BIOS_BREAK       DB    1 DUP(?)    ; BIT >=1 IF BREAK KEY HAS BEEN BIT
        ??      C
          ]     C
0072 01 [       C RESET_FLAG       DW    1 DUP(?)    ; WORD=1234H IF KEYBOARD RESET UNDERWAY
        ????    C
          ]     C
                C PAGE
                C ;-----------------------
                C ; CARD FILE DATA AREAS:
                C ;-----------------------
0074 01 [       C DISK_STATUS      DB    1 DUP(?)
        ??      C
          ]     C
0075 01 [       C HF_NUM           DB    1 DUP(?)
        ??      C
          ]     C
0076 01 [       C CONTROL_BYTE     DB    1 DUP(?)
        ??      C
          ]     C
0077 01 [       C PORT_OFF         DB    1 DUP(?)
        ??      C
          ]     C
                C ;-----------------------------------
                C ; PRINTER AND RS232 TIME-OUT VARIABLES:
                C ;-----------------------------------
0078 04 [       C PRINT_TIM_OUT    DB    4 DUP(?)
        ??      C
          ]     C
007C 04 [       C RS232_TIM_OUT    DB    4 DUP(?)
        ??      C
          ]     C
                C ;-----------------------------
                C ; ADDITIONAL KEYBOARD DATA AREA:
                C ;-----------------------------
0080 01 [       C BUFFER_START     DW    1 DUP(?)
        ????    C
          ]     C
0082 01 [       C BUFFER_END       DW    1 DUP(?)
        ????    C
          ]     C
```

```
UNIT BIOS;

INTERFACE

CONST
    RightShift  = $01;   LeftShift   = $02;   CtrlShift  = $04;
    AltShift    = $08;   ScrollState = $10;   NumState   = $20;
    CapsState   = $40;   InsState    = $80;
    ScrollShift = $10;   NumShift    = $20;   CapsShift  = $40;
    InsShift    = $80;

    FloppyFlag = $01;     InitVideoFld = $30;  nFloppyFld = $C0;
    nRS232Fld  = $0E00;   GameportFlag = $1000;
   nPrinterFld = $C000;

TYPE
   BiosDataType = record
      RS232Base,PrinterBase                : array[1..4] of word;
      EquipFlag                            : word;
      MfgTst                               : byte;
      MemorySize                           : word;
      MfgErrFlag                           : word;
      KbFlag,KbFlag1                       : byte;
      AltInput                             : byte;
      BufferHead,BufferTail                : word;
      KbBuffer                             : array[1..16] of word;
      SeekStatus,MotorStatus,MotorCount    : byte;
      DisketteStatus                       : byte;
      NECStatus                            : array[1..7] of byte;
      CrtMode                              : byte;
      CrtCols,CrtLen,CrtStart              : word;
      CursorPosn                           : array[1..8] of word;
      CursorMode                           : word;
      ActivePage                           : byte;
      Addr6845                             : word;
      CrtModeSet,CrtPalette                : byte;
      IoRomInit,IoRomSeg                   : word;
      IntrFlag                             : byte;
      TimerLow,TimerHigh                   : word;
      TimerOfl                             : byte;
      BiosBreak                            : byte;
      ResetFlag                            : word;
      DiskStatus1,HfNum                    : byte;
      ControlByte,PortOff                  : byte;
      PrintTimOut                          : array[1..4] of byte;
      RS232TimOut                          : array[1..4] of byte;
      BufferStart,BufferEnd                : word;
   end;

VAR
   BiosData  : BiosDataType          absolute $0040:$0000;
   EGA       : boolean               absolute $0040:$0087;
   BiosMark  : array[1..8] of char   absolute $F000:$FFF5;
   MachineID : byte                  absolute $F000:$FFFE;

Function MachineType:string;
Function InitVideoMode:string;
Function VideoMode:string;
Function Adapter:string;
Function RomDate:string;
Function nFloppy:byte;
Function nRS232:byte;
Function nPrinter:byte;

IMPLEMENTATION

Function BitField(bdata,field:word):word;
var i,bitrun : word;
begin
   i:=0;
   bitrun:=1;
   while (field and bitrun)=0 do begin
      inc(i);
      bitrun:=bitrun shl 1;
   end;
   BitField:=(bdata and field) shr i;
end;

Function MachineType:string;
begin
```

รูปที่ 2 (ต่อ)

```
   case MachineID of
      $FF : MachineType:='STANDARD PC';
      $FE : MachineType:='PC/XT';
      $FD : MachineType:='PCjr';
      $FC : MachineType:='PC/AT';
      $F9 : MachineType:='PC CONVERTIBLE';
      else
         MachineType:='Unknow';
   end;
end;

Function InitVideoMode:string;
var  initVmode : byte;
begin
   initVmode:=BitField(BiosData.EquipFlag,InitVideoFld);
   case initVmode of
      0 : InitVideoMode:='Unknow';
      1 : InitVideoMode:='80x25 color';
      2 : InitVideoMode:='40x25 color';
      3 : InitVideoMode:='80x25 monochrome';
      else
         InitVideoMode:='Unknow';
   end;
end;

Function VideoMode:string;
begin
   case BiosData.CrtMode of
      $00 : VideoMode:='40x25 B&W text,color adapter';
      $01 : VideoMode:='40x25 color text';
      $02 : VideoMode:='80x25 B&W text,color adapter';
      $03 : VideoMode:='80x25 color text';
      $04 : VideoMode:='320x200 4-color graphics (CGA)';
      $05 : VideoMode:=
            '320x200 4-color graphics,color burst off (CGA)';
      $06 : VideoMode:='640x200 monochrome graphics (CGA)';
      $07 : if EGA then
               VideoMode:='80x25 EGA B&W text'
            else
               VideoMode:='80x25 monochrome text';
      $08 : VideoMode:='160x200 16-color graphics (PCjr)';
      $09 : VideoMode:='320x200 16-color graphics (PCjr)';
      $0A : VideoMode:='640x200 4-color graphics (PCjr)';
      $0D : VideoMode:='320x200 16-color graphics (EGA)';
      $0E : VideoMode:='640x200 16-color graphics (EGA)';
      $0F : VideoMode:='640x350 monochrome graphics (EGA)';
      $10 : VideoMode:='640x350 4 or 16-color graphics (EGA)';
      $11 : VideoMode:='640x480 2-color graphics (VGA)';
      $12 : VideoMode:='640x480 16-color graphics (VGA)';
      $13 : VideoMode:='320x200 256-color graphics (VGA)';
      else
         VideoMode:='Unknow';
   end;
end;

Function Adapter:string;
var Vmode  : byte;
begin
   Vmode:=BiosData.CrtMode;
   if EGA then Adapter:='EGA'
   else if Vmode=2 then
      Adapter:='Compaq'
   else if Vmode=3 then
      Adapter:='CGA'
   else if (Vmode=7) and (BiosData.CrtLen>4096) then
      Adapter:='Herculess'
   else if Vmode=7 then
      Adapter:='Monochrome'
   else
      Adapter:='Unknow';
end;

Function RomDate:string;
var temps : string;
    i     : integer;
begin
   temps:='';
   for i:=1 to 8 do
      temps:=temps+BiosMark[i];
   RomDate:=temps;
```

รูปที่ 2 (ต่อ)

```
161: end;
162:
163: Function nFloppy:byte;
164: begin
165:    if (BiosData.EquipFlag and FloppyFlag)<>0 then
166:       nFloppy:=BitField(BiosData.EquipFlag,nFloppyFld)+1
167:    else
168:       nFloppy:=0;
169: end;
170:
171: Function nRS232:byte;
172: begin
173:    nRS232:=BitField(BiosData.EquipFlag,nRS232Fld);
174: end;
175:
176: Function nPrinter:byte;
177: begin
178:    nPrinter:=BitField(BiosData.EquipFlag,nPrinterFld);
179: end;
180:
181: END.
```

RS232Base เก็บแอดเดรสของพอร์ตอนุกรม มีที่สำคัญเก็บสูงสุด 4 ตัว

PrinterBase เก็บแอดเดรสของพอร์ตขนาน มีที่สำหรับเก็บสูงสุด 4 ตัว

EquipFlag เก็บค่าในลักษณะของ bitfields แต่ละฟิลด์มีค่าดังต่อไปนี้

บิต 0 : มีค่าเป็น 1 ถ้ามีฟลอปปี้ไดรฟ์อยู่

บิต 4,5 : แสดงโหมดเริ่มต้นจากจอภาพเมื่อเปิดเครื่อง

1 = จอสี 80 คอลัมน์ 25 แถว

2 = จอสี 40 คอลัมน์ 25 แถว

3 = จอโมโนโครม 80 คอลัมน์ 25 แถว

บิต 6,7 : แสดงจำนวนของฟลอปปี้ดิสค์ไดรฟ์ โดยที่ค่า 0 หมายถึงมี 1 ไดรฟ์ และ 1 หมายถึง 2 ไดรฟ์ตามลำดับ

บิต 9-11 : แสดงจำนวนพอร์ตอนุกรมที่ต่อได้ ซึ่งค่าของแอดเดรสของพอร์ตอนุกรมจะเก็บไว้ใน RS232Base ตามที่กล่าวมาแล้ว

บิต 12 : ถูกเซตเป็น 1 เมื่อต่อเกมส์พอร์ตได้

บิต 14-15 : แสดงจำนวนพอร์ตขนานที่ต่อได้ ซึ่งค่าแอดเดรสของพอร์ตขนานจะเก็บไว้ใน PrinterBase ตามที่กล่าวมาแล้ว

ค่าต่าง ๆ ใน EquipFlag เราจะใช้ตัวแปรในบรรทัดที่ 12-14 ช่วยในการอ่าน ตัวแปรที่สร้างขึ้นดังกล่าวเป็นค่าใน EquipFlag เมื่อ bitfield ที่ต้องการอ่านมีค่าเป็น 1 ทุกบิต ส่วนบิตอื่นมีค่าเป็นศูนย์ ดังนั้นเมื่อนำมา and กับค่าของ EquipFlag ที่อ่านได้จริงแล้ว shr เท่ากับตำแหน่งของ bitfield ตัวขวาสุดก็จะได้ค่าที่ต้องการ เช่น สมมติค่าของ EquipFlag = $4461 (100010001100001) เมื่อต้องการอ่านค่าของจำนวนฟลอปปี้ดิสค์ไดรฟ์ซึ่งเราใช้ตัวแปร nFloppyFld ในบรรทัดที่ 12 ช่วยในการอ่าน เมื่อนำ nFloppyFld ซึ่งเท่ากับ $CO(11000000) มา and กับ EquipFlag ได้ค่าเท่ากับ $40 (01000000) และทำการ shr เท่ากับ 6 ซึ่งเป็นตำแหน่งของ bitfield ซึ่งแสดงจำนวนของฟลอปปี้ดิสค์ไดรฟ์ได้เท่ากับ 1 ซึ่งหมายถึงมีจำนวนฟลอปปี้ดิสค์ไดรฟ์เท่ากับ 2 ไดรฟ์

MemorySize เก็บค่าจำนวนของหน่วยความจำหลักเป็นกิโลไบต์

KbFlag เก็บค่าซึ่งแสดงสถานภาพของคีย์ บิตต่าง ๆ มีค่าดังนี้

บิต 0 : ถูกเซตเมื่อแคร่ขวาถูกกดอยู่ (Right Shift key)

บิต 1 : ถูกเซตเมื่อแคร่ซ้ายถูกกดอยู่

บิต 2 : ถูกเซตเมื่อคีย์ Ctrl ถูกกดอยู่

บิต 3 : ถูกเซตเมื่อคีย์ Alt ถูกกดอยู่

บิต 4 : ถูกเซตเมื่อ Scroll Lock อยู่ในสถานะ on

บิต 5 : ถูกเซตเมื่อ Num Lock อยู่ในสถานะ on

บิต 6 : ถูกเซตเมื่อ Caps Lock อยู่ในสถานะ on

บิต 7 : ถูกเซตเมื่อ Insert อยู่ในสถานะ on

การอ่านค่าของบิตต่าง ๆ จะใช้ตัวแปรที่อยู่ในบรรทัด 6-8 เป็นตัวช่วยในการอ่านโดยนำมา and กับค่าของ KbFlag ถ้าไม่เท่ากับ 0 แสดงว่าบิตที่อ่านถูกเซต โดยใช้ RightShift ช่วยอ่านบิต 0 ซึ่งแสดงสถานะการกดคีย์ของแคร่ขวา เป็นต้น

KbFlag1 เก็บค่าซึ่งแสดงสถานภาพของคีย์ต่าง ๆ ดังนี้

บิต 4 : ถูกเซตเมื่อคีย์ Scroll

Lock ถูกกดอยู่

บิต 5 : ถูกเซตเมื่อคีย์ Num Lock ถูกกดอยู่

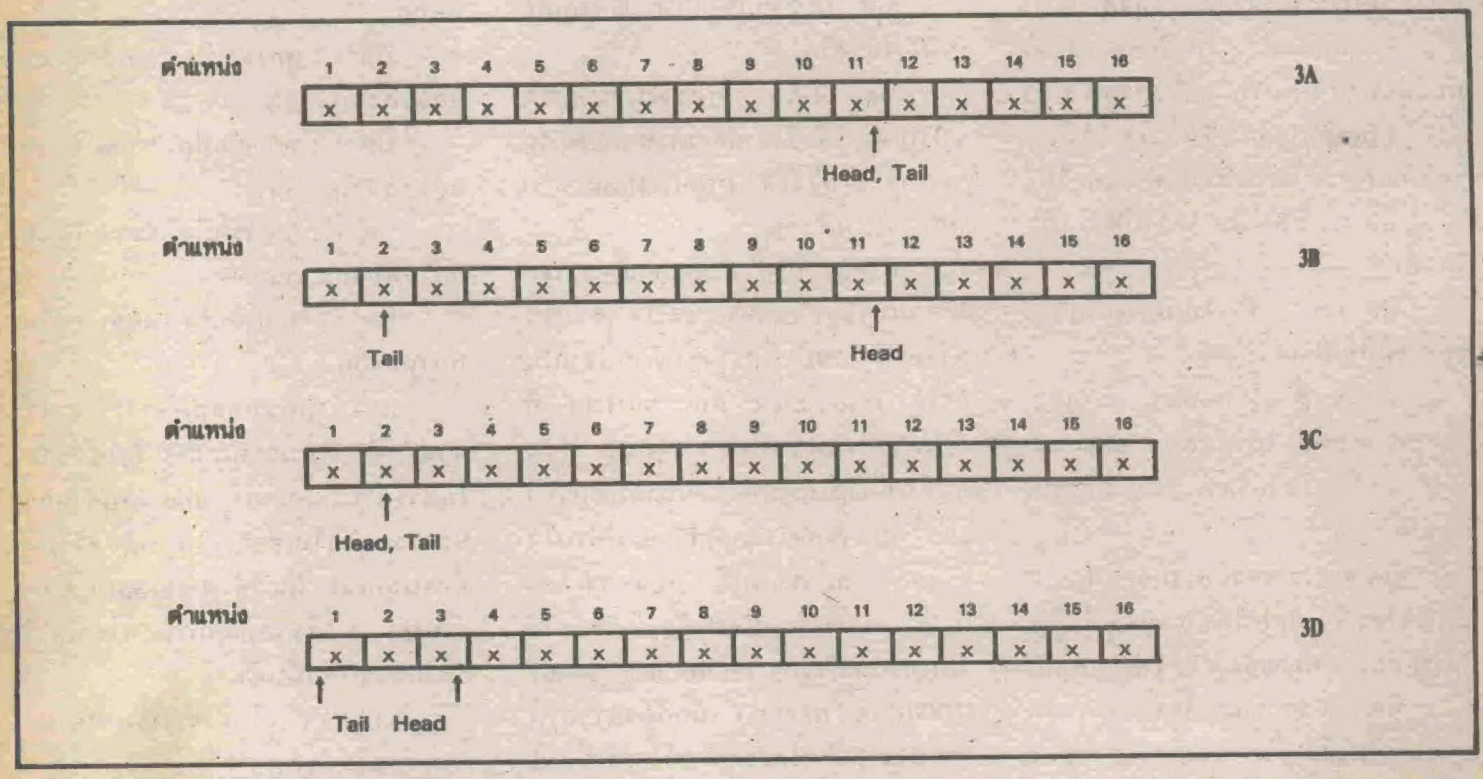
บิต 6 : ถูกเซตเมื่อคีย์ Caps Lock ถูกกดอยู่

บิต 7 : ถูกเซตเมื่อคีย์ Ins ถูกกดอยู่

เช่นเดียวกับการอ่านค่าของ KbFlag ตัวแปรที่ช่วยในการอ่านค่าของ KbFlag1 แสดงในบรรทัดที่ 9-10

AltInput ดังที่ทราบกันแล้วว่าถ้าเรากดคีย์ Alt ค้างไว้ แล้วกดคีย์ตัวเลขทางแป้น Num Lock ตามค่ารหัสแอสกีของตัวอักษรเราก็จะได้ตัวอักษรนั้น เช่น รหัสแอสกีของ 'A' คือ 65 ถ้าเรากดคีย์ Alt ค้างไว้แล้วกดคีย์เลข 6 ตามด้วยคีย์เลข 5 ของแป้นคีย์บอร์ดทาง Num Lock เราก็จะได้อักษร 'A' ในขณะที่เรากดตัวเลขแต่ละตัวค่าของ AltInput ก็จะเปลี่ยนไป เช่น เมื่อเรากดคีย์ '6' เป็นตัวแรกค่าของ AltInp ก็จะเท่ากับ 6 เมื่อเรากดคีย์ '5' เป็นตัวที่สองค่าของ AltInput ก็จะเป็น 65 ดังนั้นเมื่อเราปล่อยคีย์ Alt เราก็จะได้อักษร 'A'

BufferHead,BufferThai,KbBuffer เป็นตัวแปรที่ใช้อ่านค่าการทำงานของระบบบัฟเฟอร์ของคีย์บอร์ด ขนาดของบัฟเฟอร์มี 16 ตำแหน่ง ทำให้เราสามารถกดคีย์ล่วงหน้าได้ 15 คีย์ซึ่งค่าของคีย์ที่กดจะถูกเก็บไว้ในตัวแปร KbBuffer ค่าของคีย์แต่ละคีย์มีขนาด 2 ไบต์ ไบต์สูงเก็บค่าสแกนโค้ด ส่วนไบต์ต่ำเก็บค่ารหัสแอสกี ซึ่ง BufferHead จะชี้ที่ตำแหน่งแอดเดรสใน KbBuffer ซึ่งอักษรที่จะถูกอ่านตัวต่อไปอยู่ และ BufferTail จะชี้ที่ตำแหน่ง address ใน KbBuffer ซึ่งอักษรที่จะถูกกดตัวต่อไปจะถูกนำมาเก็บ สมมติว่าเริ่มต้น BufferHead ชี้อยู่ที่ KbBuffer[14] และ BufferTail ชี้อยู่ที่ KbBuffer[14] เช่นกันดังรูปที่ 3A นั่นหมายความว่าตอนนี้คีย์บอร์ดบัฟเฟอร์ว่างอยู่ไม่มีคีย์ใดกดล่วงหน้าไว้ก่อน แล้วต่อมามีการกดคีย์ล่วงหน้า 4 คีย์ ในขณะที่ยังไม่มีการอ่านคีย์เนื่องจากกำลังประมวลผลอยู่ ซึ่งจะมีผลทำให้ BufferTail เลื่อนออกไป 4 ตำแหน่งจากตำแหน่งที่ 14 ของบัฟเฟอร์ไปยังตำแหน่งของ KbBuffer[2] เนื่องจากมีสูงสุดแค่ 16 ตำแหน่ง ดังรูปที่ 3B เมื่อกดคีย์ล่วงหน้า 4 คีย์แล้ว ปรากฏว่าเกิดมีการอ่านคีย์ทั้ง 4 คีย์ทำให้ BufferHead เลื่อนมาตามจำนวนของคีย์ที่อ่านมาอยู่ดังตำแหน่งที่ 2 ดังรูปที่ 3C ต่อมาเกิดการกดคีย์ล่วงหน้าอีก 15 คีย์ทำให้ BufferTail วนมาอยู่ยัง



รูปที่ 3 ตำแหน่งต่าง ๆ ของ Buffer Head และ Buffer Tail

ตำแหน่ง 1 ดังรูปที่ 3D ในตอนนี้บัฟเฟอร์ของคีย์บอร์ดจะเต็มแล้ว ถ้าเรากดคีย์ล่วงหน้าอีก BuferTail จะไม่เลื่อน แต่จะมีเสียงเตือนดังขึ้น เนื่องจากจะทำให้ตำแหน่งของ BufferTail เท่ากับตำแหน่งของ BufferHead ซึ่งหมายถึงบัฟเฟอร์ว่างเปล่า สังเกตว่า KbBuffer มี 16 ตำแหน่งสามารถรับคีย์ล่วงหน้าได้ 15 คีย์ อย่างไรก็ตามเมื่อเรากดคีย์ล่วงหน้าในขณะที่โปรแกรมที่ทำงานอยู่ยังไม่มีการอ่านคีย์ หรือยังอ่านคีย์ที่กดไปได้ไม่ทั้งหมด เราสามารถยกเลิกคีย์ที่กดล่วงหน้าได้ด้วยการกด Ctrl-Break แต่ทั้งนี้โปรแกรมดังกล่าวจะต้องเซต Break off ถ้าเป็นโปรแกรมที่เขียนด้วยเทอร์โบปาสคาลก็ต้องใช้ Check-Break:=False แต่ต้องไม่ทำงานอยู่ใน integrated development environment (IDE)

CrtMode แสดงโหมดของจอภาพที่ใช้อยู่ ค่าของ CrtMode มีความหมายดังนี้

$00 : 40×25 B&W text,color adapter
$01 : 40×25 color text
$02 : 80×25 B&W text,color adapter
$03 : 80×25 color text
$04 : 320×200 4-color graphics (CGA)
$05 : 320×200 4-color graphics, color burst off (CGA)
$06 : 640×200 monochrome graphics (CGA)
$07 : 80×25 EGA B&W text or 80×25 monochrome text
$08 : 160×200 16-color graphics (PCjr)
$09 : 320×200 16-color graphics (PCjr)
$0A : 640×200 4-color graphics (PCjr)
$0D : 320×200 16-color graphics (EGA)
$0E : 640×200 16-color graphics (EGA)
$0F : 640×350 monochrome graphics (EGA)
$10 : 640×350 4 or 16 color graphics (EGA)
$11 : 640×480 2-color graphics (VGA)
$12 : 640×480 16-color graphics (VGA)
$13 : 320×200 256-color graphics (VGA)

CrtCols แสดงจำนวนคอลัมน์ของจอภาพในโหมดที่เซตอยู่

CrtLen แสดงขนาดหน่วยความจำของจอภาพเป็นจำนวนไบต์

CrtStart แสดงตำแหน่งเริ่มแรกของจอภาพ

CursorPosn แสดงตำแหน่งของเคอร์เซอร์บนจอภาพแต่ละหน้า CursorPosn[1] แสดงตำแหน่งเคอร์เซอร์ในหน้าที่ 0 ของจอภาพ โดยไบต์ต่ำแสดงตำแหน่ง x และไบต์สูงแสดงตำแหน่ง y ตำแหน่งของเคอร์เซอร์ที่เก็บจะเริ่มที่ (0,0) มุมจอภาพซ้ายมือด้านบน จำนวนหน้าของจอภาพมีได้สูงสุด 8 หน้าตั้งแต่หน้าที่ 0 ถึงหน้าที่ 7

CursorMode แสดงโหมดของเคอร์เซอร์ ซึ่งก็คือขนาดของเคอร์เซอร์จะเริ่มที่สแกนไลน์ที่เท่าไรถึงสแกนไลน์ที่เท่าไร ไบต์สูงเก็บค่าสแกนไลน์ที่เริ่มต้น ไบต์ต่ำเก็บค่าสแกนไลน์ที่สิ้นสุด

ActivePage ดังที่กล่าวแล้วว่าจอภาพมีอยู่ด้วยกันหลายหน้า ค่าของ ActivePage จะเก็บค่าที่แสดงหน้าของจอภาพที่แสดงอยู่

Addr6845 แสดงค่าพอร์ตของวีดิโออะแดปเตอร์

CrtPalette ค่าของ palette ที่ใช้อยู่

BufferStart,BufferEnd แสดงค่าแอดเดรสของคีย์บอร์ดบัฟเฟอร์ทั้ง 16 ตำแหน่ง BufferStart เก็บค่าแอดเดรสของตำแหน่งที่ 1 BufferEnd จะชี้ที่แอดเดรสเหนือตำแหน่งสุดท้ายของคีย์บอร์ดบัฟเฟอร์

หลังจากที่เราได้สร้างแบบของข้อมูลให้มีลักษณะเหมือนกับรอมไบออสดาต้าเอเรียส์แล้ว ขั้นต่อไปคือสร้างตัวแปรให้เป็นประเภท BiosDataType และต้องให้ตำแหน่งของตัวแปรที่สร้างขึ้นอยู่ตำแหน่งเดียวกันกับรอมไบออสดาต้าเอเรียส์ที่อยู่ในหน่วยความจำด้วยบรรทัดที่ 51 ในรูปที่ 2 แสดงการสร้างตัวแปรดังกล่าวโดยให้ชื่อว่า BiosData โดยใช้ absolute ช่วยกำหนดตำแหน่งในหน่วยความจำที่ต้องการ ซึ่งตำแหน่งของรอมไบออสดาต้าเอเรียส์เริ่มต้นที่ $0040:000 เมื่อเราสร้างตัวแปรได้แล้ว เราก็สามารถอ่านค่าต่าง ๆ ของรอมไบออสดาต้าเอเรียส์จากตัวแปรที่สร้างขึ้น เช่น เราต้องการอ่านค่าโหมดของจอภาพ (CrtMode) เราก็อ่านจาก BiosData.CrtMode ค่าอื่นก็อ่านได้ในลักษณะเดียวกัน

นอกจากรอมไบออสดาต้าเอเรียส์แล้ว ยังมีตำแหน่งในหน่วยความจำอื่นอีกที่น่าสนใจได้แก่

$0040:$0087 ข้อมูลในตำแหน่งนี้จะถูกเซตเป็น 0 ถ้าจอภาพที่ใช้อยู่ไม่ใช่จออีจีเอหรือวีจีเอ ในกรณีอื่นจะถูกเซตเป็นค่าอื่น ดังนั้นเราจึงสร้างตัวแปร EGA ให้เป็นแบบบูลีนที่ตำแหน่งหน่วยความจำนี้ ดังแสดงในรูปที่ 2 บรรทัดที่ 52

$F000:$FFF5 เก็บวันที่ของรอมไบออสไว้เป็นตัวอักษรในลักษณะ mm/

dd/yy วันที่ดังกล่าวเป็นตัวบอกเวอร์ชันของรอมตัวแปร BiosMark จึงถูกสร้างขึ้นเพื่ออ่านค่านี้

$F000:$FFFE เก็บค่าซึ่งแสดงประเภทของเครื่อง ซึ่งเราสร้างตัวแปร MachineID ไว้อ่านค่าที่ตำแหน่งนี้ ค่าต่างที่ใช้มีดังนี้

$FF : Standard PC
$FE : XT
$FD : PCjr
$FC : AT,XT286,PS/2-50, PS/2-60
$FA : PS/2-30
$F9 : PC Convertible
$F8 : PS/2-80

นอกจากตัวแปรต่าง ๆ ที่สร้างขึ้นแล้วยูนิต BIOS ยังมีฟังก์ชันไว้แสดงหรืออ่านค่าใน ROM BIOS DATA AREAS ที่ต้องมีการแปลความหมาย หรืออยู่ในลักษณะที่ต้องอ่านเป็น BitField เพื่อให้สะดวกแก่การใช้งาน จากส่วนของ IMPLEMENTATION ของยูนิตไบออสฟังก์ชันต่าง ๆ ให้ค่าดังนี้

BitField(bdata,field) ให้ค่าของ bitfield ที่อ่าน bdata เป็นข้อมูลที่ต้องการอ่านค่าเป็น bitfield ส่วนฟิลด์เป็นตัวช่วยอ่านที่ถูกเซตบิตเฉพาะฟิลด์ที่จะอ่าน เช่น เราต้องการอ่านค่าของ EquipFlag โดยอ่านเป็น bitfield ของบิต 6,7 เราก็เรียกใช้ BitField(EquipFlag,$CO) ซึ่งค่าของ $CO ในฐาน 2 ก็คือ 11000000

MachineType อ่านค่าของตัวแปร MachineID ที่สร้างขึ้นที่ตำแหน่ง $F000:$FFFE แล้วให้ค่าเป็นสตริงซึ่งข้อความแสดงประเภทของเครื่อง

InitVideoMode อ่านค่าในบิต 4,5 ของ BiosData.EquipFlag ซึ่งแสดงโหมดเริ่มต้นของจอภาพเมื่อเปิดเครื่อง แล้วให้ค่าเป็นสตริงซึ่งแสดงข้อความที่บอกโหมดดังกล่าว

VideoMode อ่านค่าของ BiosData.CrtMode แล้วให้ข้อความเป็นสตริงแสดงโหมดของจอภาพที่ถูกเซตอยู่

Adapter ให้ข้อความแสดงประเภทอะแดปเตอร์ของจอภาพที่ใช้

RomDate อ่านค่าจากตัวแปร BiosMark แล้วเปลี่ยนเป็นสตริงแสดงวันที่ของรอม

nFloppy ให้ค่าแสดงจำนวนของฟลอปปี้ดิสก์ไดรฟ์

nRS232 ให้ค่าแสดงจำนวนของพอร์ตอนุกรม

nPrinter ให้ค่าแสดงจำนวนของพอร์ตขนาน

ตัวอย่างโปรแกรม

จากยูนิตไบออส เราสามารถนำไปใช้ประโยชน์ต่างได้มากมาย โปรแกรม ShowBios ในรูปที่ 4 เป็นโปรแกรมที่เขียนขึ้นเพื่อเป็นตัวอย่างในการอ่านค่าต่าง ๆ มาใช้ตัวโปรแกรมสามารถแยกออกได้เป็นสามส่วนด้วยกัน คือ

รูปที่ 4 SHOWBIOS.PAS

```
 1: PROGRAM ShowBios;
 2: uses crt,bios;
 3: {-------------------------------------------------------------}
 4: {                        Utility Part                         }
 5: {-------------------------------------------------------------}
 6: Function HexStr(num:word):string;
 7: const HexDigit : string[16] = '0123456789ABCDEF';
 8: var   i,mask   : integer;
 9:       temps    : string;
10: begin
11:    temps:='';
12:    for i:=1 to sizeof(num)*2 do begin
13:       mask:=num and $000F;
14:       num:=num shr 4;
15:       temps:=HexDigit[mask+1]+temps;
16:    end;
17:    HexStr:='$'+temps;
18: end;
19:
20: Procedure writeline(x,y,color:integer;s:string);
21: begin
22:    gotoxy(x,y);
23:    textcolor(color);
24:    write(s);
25: end;
26:
27: Procedure writenum(x,y,color,num:integer);
28: begin
29:    gotoxy(x,y);
30:    textcolor(color);
```

```
31:    write(num);
32: end;
33:
34: Procedure writeline2(x,y,fg,bg:integer;s:string);
35: begin
36:    gotoxy(x,y);
37:    textcolor(fg);
38:    textbackground(bg);
39:    write(s);
40:    textbackground(Black);
41: end;
42: {-------------------------------------------------------------}
43: {                    Sub Program Part                         }
44: {-------------------------------------------------------------}
45: Procedure SystemInformation;
46: const   maxshow = 13;
47:         TextShow : array[1..maxshow] of string =
48:                  (  'ROM BIOS DATE       :',
49:                     'MACHINE TYPE        :',
50:                     'MAIN MEMORY         :      KB',
51:                     'FLOPPY DISK DRIVES  :',
52:                     'INITIAL VIDEO MODE  :',
53:                     'VIDEO BUFFER SIZE   :      BYTES',
54:                     'VIDEO MODE          :',
55:                     'ACTIVE VIDEO PAGE   :',
56:                     'VIDEO ADAPTER PORT  :',
57:                     'ADAPTER             :',
58:                     'CURSOR MODE         :',
59:                     'SERIAL PORT         :',
60:                     'PARALLEL PORT       :'  );
61: var loops : integer;
62:     ch    : char;
63: begin
64:    clrscr;
65:    for loops:=1 to maxshow do
66:       writeline(20,5+loops,Green,TextShow[loops]);
67:    writeline(41,5+1,Yellow,RomDate);
68:    writeline(41,5+2,Yellow,MachineType);
69:    writenum(41,5+3,Yellow,BiosData.MemorySize);
70:    writenum(41,5+4,Yellow,nFloppy);
```

```
71:    writeline(41,5+5,Yellow,InitVideoMode);
72:    writenum(41,5+6,Yellow,BiosData.CrtLen);
73:    writeline(41,5+7,Yellow,VideoMode);
74:    writenum(41,5+8,Yellow,BiosData.ActivePage);
75:    writeline(41,5+9,Yellow,HexStr(BiosData.Adk  $945));
76:    writeline(41,5+10,Yellow,Adapter);
77:    writeline(41,5+11,Yellow,HexStr(BiosData.CursorMode));
78:    gotoxy(41,5+12);
79:    for loops:=1 to nRS232 do begin
80:       write(HexStr(BiosData.RS232Base[loops]));
81:       if loops<nRS232 then write(',');
82:    end;
83:    gotoxy(41,5+13);
84:    for loops:=1 to nPrinter do begin
85:       write(HexStr(BiosData.PrinterBase[loops]));
86:       if loops<nPrinter then write(',');
87:    end;
88:    writeline(25,25,Green+Blink,'Strike <Esc> for continue');
89:    repeat
90:       ch:=readkey;
91:    until ch=$27;
92: end;
93:
94: Procedure KeyboardStatus;
95:
96:       Function Check01(kbdata,field:byte):byte;
97:       begin
98:          if kbdata and field=0 then
99:             Check01:=0
100:         else
101:            Check01:=1;
102:      end;
103:
104: const  state : array[0..1] of string = ( 'off','on ');
105:       Down  : array[0..1] of string = ( 'Not Down','Down     ');
106:
107:       maxshow = 9;
108:       textshow : array[1..maxshow] of string  =
109:                ( 'Right Shift is  .',
110:                  'Left Shift is   :',
```

รูปที่ 4 (ต่อ)

```
111:                    'Ctrl is        :',
112:                    'Alternate is   :',
113:                    'ScrollLock is  :       and :',
114:                    'NumLock is     :       and :',
115:                    'CapsLock  is   :       and :',
116:                    'Insert is      :       and :',
117:                    'Alt Input is   :'  );
118: var  kb_,kbl_,AltInp_ : byte;
119:      loopk           : integer;
120:      ch              : char;
121: begin
122:    clrscr;
123:    writeline(25,25,Green+Blink,'Strike <Esc> for continue');
124:    for loopk:=1 to maxshow do
125:       writeline(20,5+loopk,Green,textshow[loopk]);
126:    kb_:=BiosData.KbFlag xor 1;
127:    kbl_:=BiosData.KbFlag1 xor 1;
128:    AltInp_:=BiosData.AltInput xor 1;
129:    repeat
130:       if keypressed then
131:          ch:=readkey;
132:       if kb_<>BiosData.KbFlag then begin
133:          kb_:=BiosData.KbFlag;
134:          writeline(37,5+1,Yellow,Down[Check01(kb_,RightShift)]);
135:          writeline(37,5+2,Yellow,Down[Check01(kb_,LeftShift)]);
136:          writeline(37,5+3,Yellow,Down[Check01(kb_,CtrlShift)]);
137:          writeline(37,5+4,Yellow,Down[Check01(kb_,AltShift)]);
138:          writeline(52,5+5,Yellow,State[Check01(kb_,ScrollState)]);
139:          writeline(52,5+6,Yellow,State[Check01(kb_,NumState)]);
140:          writeline(52,5+7,Yellow,State[Check01(kb_,CapsState)]);
141:          writeline(52,5+8,Yellow,State[Check01(kb_,InsState)]);
142:       end;
143:       if kbl_<>BiosData.KbFlag1 then begin
144:          kbl_:=BiosData.KbFlag1;
145:          writeline(37,5+5,Yellow,Down[Check01(kbl_,ScrollShift)]);
146:          writeline(37,5+6,Yellow,Down[Check01(kbl_,NumShift)]);
147:          writeline(37,5+7,Yellow,Down[Check01(kbl_,CapsShift)]);
148:          writeline(37,5+8,Yellow,Down[Check01(kbl_,InsShift)]);
149:       end;
150:       if AltInp_<>BiosData.AltInput then begin
151:          AltInp_:=BiosData.AltInput;
152:          gotoxy(37,5+9);
153:          write(AltInp_);
154:          clreol;
155:       end;
156:    until ch=#27;
157: end;
158:
159: Procedure KeyboardBufferDemo;
160: const Seg=$40;
161: var loopk          : integer;
162:     tail_,KBdata   : word;
163:     ch             : char;
164:     KBdataP        : ^word;
165:     KBdataV        : word absolute KBdataP;
166: begin
167:    clrscr;
168:    writeline(1,3,Yellow,' Mark    Address    Data    Character');
169:    with BiosData do
170:    begin
171:       for loopk:=1 to 16 do begin
172:          if addr(KbBuffer[loopk])=Ptr(Seg,BufferHead) then
173:             writeline(1,4+loopk,Green,'Head-->');
174:          KBdataP:=addr(KbBuffer[loopk]);
175:          writeline(11,4+loopk,Green,HexStr(KBdataV));
176:       end;
177:       tail_:=BufferTail xor 1;
178:       repeat
179:          if tail_<>BufferTail then begin
180:             tail_:=BufferTail;
181:             for loopk:=1 to 16 do begin
182:                if addr(KbBuffer[loopk])=Ptr(Seg,BufferHead) then
183:                   writeline(1,4+loopk,Green,'Head-->');
184:                if Addr(KbBuffer[loopk])=Ptr(Seg,BufferTail) then
185:                begin
186:                   if BufferTail<>BufferHead then
187:                      writeline(1,4+loopk,Green,'Tail-->')
188:                end else begin
189:                   if addr(KbBuffer[loopk])<>Ptr(Seg,BufferHead)
190:                   then
```

```
191:                    writeline(1,4+loopk,Green,'        ');
192:                end;
193:                KBdata:=KbBuffer[loopk];
194:                writeline(21,4+loopk,Green,HexStr(KBdata));
195:                writeline(33,4+loopk,Green,chr(KBdata));
196:             end;
197:          end;
198:       until (BufferHead-tail_=2) or
199:             (tail_-BufferHead=30);
200:       loopk:=0;
201:       writeline(1,22,Green,'Character read      :');
202:       textcolor(Yellow);
203:       while keypressed do begin
204:          ch:=readkey;
205:          write(char(ch));
206:          if ch<>#0 then
207:             inc(loopk);
208:       end;
209:       writeline(1,23,Green,'Number of character :');
210:       textcolor(Yellow);
211:       write(loopk);
212:       writeline(1,25,Green+Blink,'Strike <Esc> for continue');
213:       repeat
214:          ch:=readkey;
215:       until ch=#27;
216:    end;
217: end;
218: {-------------------------------------------------------------}
219: {                    Main Program Part                        }
220: {-------------------------------------------------------------}
221: BEGIN
222:    CheckBreak:=False;
223:    SystemInformation;
224:    KeyboardStatus;
225:    KeyBoardBufferDemo;
226: END.
```

รูปที่ 4 (ต่อ)

Utility Part เริ่มจากบรรทัดที่ 3-41 มีฟังก์ชันและโพรซีเดอร์ต่าง ๆ ดังนี้

- HexStr(num:word) เป็นฟังก์ชันให้ค่าสตริงที่แสดงค่าในฐาน 16 ของ num โดยมีเครื่องหมาย $ นำหน้า

- writeline(x,y,color:integer;s: string) แสดงสตริง s บนจอภาพที่ตำแหน่ง (x,y) ด้วยสี color

- writenum(x,y,color,num: integer) แสดงค่าของ num บนจอภาพในตำแหน่ง (x,y) ด้วยสีคัลเลอร์

- writeline2(x,y,fg,bg:integer; string) แสดงสตริง s บนจอภาพที่ตำแหน่ง (x,y) ด้วยสี fg และสีของพื้นเป็น bg

Sub Program Part เริ่มจากบรรทัด 42-217 มีโพรซีเดอร์ ดังนี้

- SystemInformation จะแสดงค่าต่าง ๆ ที่เกี่ยวข้องกับสภาพการทำงานของระบบ เช่น ประเภทเครื่อง,จำนวนพอร์ตอนุกรม เป็นต้น

- KeyboardStatus จะแสดงค่าของคีย์ล็อคต่าง ๆ ว่ากดอยู่หรือไม่ ได้แก่พวก Ins,Ctrl,Alt,Caps,Scroll ว่ากดค้างอยู่หรือไม่ หรือโหมดการทำงานอยู่ในสภาพ on หรือ off เช่น โหมดของ Ins,Caps ค่าต่าง ๆ จะอ่านจากบิตต่าง ๆ ใน BiosData.KbFlag, BiosData.KbFlag1 สังเกตในลูปตั้งแต่

บรรทัดที่ 129-156 จะเห็นว่าโดยปกติจะไม่อ่านคีย์ นอกจากมีการแตะคีย์ในบรรทัดที่ 130 ฟังก์ชันย่อย Check01 จะเป็นตัวตรวจสอบค่าของบิตที่อ่านว่าถูกเซตหรือไม่ สังเกตการตรวจสอบด้วย if ในบรรทัดที่ 132,143 และ 150 ซึ่งจะตรวจสอบค่าของ BiosData.KbFlag,BiosData.KbFlag1 และ BiosData.AltInput ตามลำดับเพื่อจะทำการแสดงค่าใหม่เมื่อค่าในตัวแปรเหล่านั้นเปลี่ยนแปลงเท่านั้น โดยเปรียบเทียบกับค่าเก่าที่จะเก็บไว้ใน kb__,kb1__, และ AltInp__ ตามลำดับ ค่าเริ่มต้นของ kb__,kb1__ และ AltInp__ จะถูกกำหนดให้มีค่าต่างจากตัวแปรที่อ่านจาก BiosData ก่อนในบรรทัดที่ 126-128 เพื่อให้มีการแสดงค่าในครั้งแรกทันทีที่เข้าในลูป repeat until เมื่อทำการรันโปรแกรมแล้ว สิ่งที่น่าสนใจสิ่งหนึ่งคือค่าของ BiosData.AltInput ซึ่งเป็นค่าที่จะเปลี่ยนไปเมื่อกดคีย์ Alt ค้างไว้แล้วกดคีย์ตัวเลข โพรซีเดอร์ KeyboardStatus จะแสดงสถานะของคีย์ต่าง ๆ ที่เปลี่ยนไปจนกว่าผู้ใช้จะกดคีย์ Esc

- KeyboardBufferDemo เป็นโพรซีเดอร์ที่แสดงการทำงานของคีย์บอร์ดบัฟเฟอร์ โดยแสดงตำแหน่งที่ชี้อยู่ของ BufferHead และ BufferTail ในตำแหน่งต่าง ๆ ของ KbBuffer รวมทั้งตำแหน่งในหน่วยความจำของข้อมูลแต่ละตัว ทำให้สามารถสังเกตการเปลี่ยนแปลงต่าง ๆ ของข้อมูลในบัฟเฟอร์ได้ ในการสังเกตการทำงานของโพรซีเดอร์นี้ควรจะคอมพายล์โปรแกรมเป็น .EXE แล้วค่อยเรียกโปรแกรมขึ้นมาทำงานเพื่อให้สามารถสังเกตการเปลี่ยนแปลงของ BufferHead และ BufferTail เมื่อกด Ctrl-Break ได้ หลังจากที่เรากดคีย์จนบัฟเฟอร์เต็มแล้ว โปรแกรมจะแสดงค่าของคีย์บอร์ดที่อ่านได้จากฟังก์ชัน ReadKey ของเทอร์โบปาสคาล จะเห็นความแตกต่างของค่าคีย์บอร์ดของ extended key เช่น ฟังก์ชันคีย์ต่าง ๆ ซึ่งฟังก์ชัน ReadKey ของเทอร์โบปาสคาลจะให้ค่าของคีย์ extended เป็นสองค่าคือ #0 และค่าแสกนโค้ด

Main Program Part เริ่มจากบรรทัด 218-226 จะเรียกใช้โพรซีเดอร์ต่าง ๆ ที่กล่าวมาแล้วใน Sub Program Part มาแสดงการทำงาน

ตัวอย่างโปรแกรม ShowBios นี้จะสามารถทำให้ผู้อ่านเห็นตัวอย่างของการนำไปใช้งานได้ชัดเจนขึ้น ค่าที่อ่านขึ้นมาแสดงนี้ไม่ได้อ่านขึ้นมาทุกค่า ค่าใน BiosData นี้ควรใช้ในการอ่านอย่างเดียวเท่านั้นไม่ควรใช้ในการเขียน เช่น การเปลี่ยนโหมดจอภาพโดยการกำหนดใน BiosData.CrtMode จะไม่ทำให้จอภาพเปลี่ยนไปอยู่ในโหมดตามที่ต้องการ แต่ก็มีผลทำให้การทำงานของระบบเปลี่ยนแปลงไป ค่าหลายค่าสามารถอ่านหรือเขียนผ่านทางอินเตอร์รัพต์ซึ่งเป็นวิธีที่ปลอดภัยกว่า โดยเฉพาะการเปลี่ยนแปลงค่าควรกระทำผ่านทางอินเตอร์รัพต์เสมอ นอกจากนั้นในเทอร์โบปาสคาลยังมีฟังก์ชันที่ทำหน้าที่ให้ค่าต่าง ๆ คล้ายกับค่าที่อ่านจาก BiosData อยู่หลายค่วยกัน เช่น การอ่านค่าโหมดของจอภาพในเท็กซ์โหมดสามารถอ่านได้จากตัวแปร LastMode การอ่านค่าโหมดของจอภาพในกราฟฟิกโหมดสามารถอ่านได้เช่นกันจากฟังก์ชัน GetGraphMode และ GetDriverName การอ่านตำแหน่งของเคอร์เซอร์สามารถอ่านได้จากฟังก์ชัน wherex และ Wherey อย่างไรก็ตามการอ่านค่าจาก BiosData คงเป็นประโยชน์ต่อท่านผู้อ่านในการพัฒนาโปรแกรมอยู่บ้าง